

Sequential Kernel Density Approximation and Its Application to Real-Time Visual Tracking

Bohyung Han, *Member, IEEE* Dorin Comaniciu, *Senior Member, IEEE* Ying Zhu, *Member, IEEE*
and Larry S. Davis, *Fellow, IEEE*

Abstract—Visual features are commonly modeled with probability density functions in computer vision problems, but current methods such as a mixture of Gaussians and kernel density estimation suffer from either the lack of flexibility, by fixing or limiting the number of Gaussian components in the mixture, or large memory requirement, by maintaining a non-parametric representation of the density. These problems are aggravated in real-time computer vision applications since density functions are required to be updated as new data becomes available. We present a novel kernel density approximation technique based on the mean-shift mode finding algorithm, and describe an efficient method to sequentially propagate the density modes over time. While the proposed density representation is memory efficient, which is typical for mixture densities, it inherits the flexibility of non-parametric methods by allowing the number of components to be variable. The accuracy and compactness of the sequential kernel density approximation technique is illustrated by both simulations and experiments. Sequential kernel density approximation is applied to on-line target appearance modeling for visual tracking, and its performance is demonstrated on a variety of videos.

Index Terms—kernel density approximation, mean-shift, mode propagation, on-line target appearance modeling, object tracking, real-time computer vision

I. INTRODUCTION

DENSITY estimation is broadly used to statistically model visual features in computer vision applications. The underlying probability density of the features can be described by a parametric (e.g., Gaussian or mixture of Gaussians) or non-parametric (e.g., histogram or kernel density-based) representation. However, these representations create a trade-off between the flexibility of the model and its data summarization property. In other words, parametric methods are simple and efficient, but have difficulty representing multi-modal density functions effectively. Also, they usually require a pre-defined parameter for the number of components, so it is hard to use parametric density functions in real-time applications, especially when there are a large number of modes in the underlying density or the number of modes is frequently changing. On the other hand, non-parametric models are very flexible and can accommodate complex densities, but require a large amount of memory for their implementation.

We present a new method to approximate a multi-modal density function with a mixture of Gaussians — Kernel Density Approximation (KDA), which was originally introduced in [15], [16]. Kernel density approximation is a flexible multi-modal density representation method since every parameter for the Gaussian mixture is determined automatically. This technique is applied to a real-time computer vision problem — on-line target appearance modeling for object tracking.

A. Related Work

A Gaussian distribution, which is the simplest density-based modeling method, is frequently used for various computer vision problems, such as background subtraction and object tracking [14], [17], [24], [32]. However, this representation cannot handle multi-modal density functions, so the accuracy of modeling is severely limited.

Mixture models based on multiple components have been utilized in numerous applications. In [22], a target appearance model based on color is constructed by a mixture of Gaussians which is replaced in each frame, and the same density representation is employed for optical flow estimation by Jepson and Black [18]. A recursive update of a Gaussian mixture model is proposed in [20], [29] for background modeling, but these methods are not flexible enough to model complex density functions since they typically require the maximum number of components in the mixture in advance. For object tracking, adaptive target appearance modeling by a 3-component mixture is described in [19], where the mixture density function is updated over time by an on-line EM algorithm. However, it is generally difficult to add or remove components in the existing adaptive mixture models in a principled way. Therefore, most real-time applications rely on models with a fixed number of mixtures [18], [19], [22] or apply ad-hoc strategies to adapt the number of mixtures in time [20], [27], [29], where the addition and deletion of a Gaussian component is highly dependent on the pre-defined threshold values. There is a more elaborated method to determine the number of components using a layer model, but it also requires the maximum number of layers as a parameter and the decision regarding the number of layers is based on an additional complex process [30].

Kernel density estimation [12] is one of the most popular non-parametric techniques to model densities, because it provides a flexible framework to represent multi-modal densities. However, its very high memory requirements and computational complexity inhibit the use of this method in real-time applications, even though there have been several attempts to reduce the computational cost [11], [33].

B. Our Approach

In contrast to previous approaches, we present a new strategy for multi-modal density approximation and its on-line learning that relies on modeling and propagation of density modes. The modes (local maxima) of a density function represent regions with higher local probability; hence, their preservation is important to maintain a low density approximation error. We represent the density as a weighted sum of Gaussians, whose number, weights, means and covariances are automatically determined. The mode locations are detected by a mode finding algorithm based on the variable-bandwidth mean-shift, and Gaussian components are

created whose means are given by mode locations. Also, the covariance of each Gaussian is derived by fitting the curvature around its mode location. For sequential adaptation, the density function can be updated at each time step to include new data into the model. Starting from the previous components of the density and the new component, we use the mean-shift algorithm to detect new modes. This procedure is performed in linear time, which we prove by amortized analysis [10]. Using this mode-based representation, the memory requirements are low, similar to all methods that use mixture densities, but we have a principled way to create and delete Gaussian components at each time step.

The proposed algorithm is able to preserve mode locations in underlying density functions and approximate well-separated Gaussian mixtures very accurately. It is not a general density approximation method, but is tuned for the types of densities often encountered in computer vision applications. In on-line computer vision applications, all the previous data is typically not available at processing time, and the preservation and propagation of major density modes is more important than an accurate approximation of the entire density function. Our approach is particularly appropriate for real-time computer vision applications such as on-line target appearance modeling for visual tracking.

The paper is organized as follows. Section II discusses kernel density approximation by mode detection based on the variable-bandwidth mean-shift. Sequential kernel density approximation using mode propagation is explained in section III. How this technique can be applied to on-line target appearance modeling for object tracking is described in section IV.

II. KERNEL DENSITY APPROXIMATION

This section explains how we approximate a density function with a mixture of Gaussians. The mean-shift mode finding algorithm [7]–[9] is presented, and a covariance estimation technique based on curvature fitting is also discussed. After that, the accuracy of the kernel density approximation is illustrated through simulation.

A. Mean-Shift Mode Finding

Suppose that we are given a density function by weighted kernel density estimation based on a Gaussian kernel. Denote by \mathbf{x}_i ($i = 1, \dots, n$) a set of means of Gaussians in R^d and by \mathbf{P}_i a symmetric positive definite $d \times d$ covariance matrix associated with the corresponding Gaussian. Let each Gaussian have a weight κ_i with $\sum_{i=1}^n \kappa_i = 1$. The sample point density estimator computed at point \mathbf{x} is given by

$$\hat{f}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^n \frac{\kappa_i}{|\mathbf{P}_i|^{1/2}} \exp\left(-\frac{1}{2}D^2(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i)\right) \quad (1)$$

where

$$D^2(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i) \equiv (\mathbf{x} - \mathbf{x}_i)^\top \mathbf{P}_i^{-1} (\mathbf{x} - \mathbf{x}_i) \quad (2)$$

is the Mahalanobis distance from \mathbf{x} to \mathbf{x}_i . The density at \mathbf{x} is obtained as the *average of Gaussian densities* centered at each data point \mathbf{x}_i and having the covariance \mathbf{P}_i .

In the underlying density function (1), the variable-bandwidth mean-shift vector at location \mathbf{x} is defined by

$$\mathbf{m}(\mathbf{x}) = \left(\sum_{i=1}^n \omega_i(\mathbf{x}) \mathbf{P}_i^{-1} \right)^{-1} \left(\sum_{i=1}^n \omega_i(\mathbf{x}) \mathbf{P}_i^{-1} \mathbf{x}_i \right) - \mathbf{x} \quad (3)$$

where the weights

$$\omega_i(\mathbf{x}) = \frac{\kappa_i |\mathbf{P}_i|^{-1/2} \exp\left(-\frac{1}{2}D^2(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i)\right)}{\sum_{i=1}^n \kappa_i |\mathbf{P}_i|^{-1/2} \exp\left(-\frac{1}{2}D^2(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i)\right)} \quad (4)$$

satisfy $\sum_{i=1}^n \omega_i(\mathbf{x}) = 1$.

It can be shown that by iteratively computing the mean-shift vector (3) and translating the location \mathbf{x} by $\mathbf{m}(\mathbf{x})$, a mode seeking algorithm is obtained which converges to a stationary point of the density in equation (1) [8]. There are three kinds of stationary points in the density function: local maxima, local minima and saddle points. We are interested in finding mode locations (local maxima) to simplify the original density function; a formal check for the maxima involves the computation of the Hessian matrix

$$\hat{\mathbf{H}}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^n \frac{\kappa_i}{|\mathbf{P}_i|^{1/2}} \exp\left(-\frac{1}{2}D^2(\mathbf{x}, \mathbf{x}_i, \mathbf{P}_i)\right) \times \mathbf{P}_i^{-1} \left((\mathbf{x}_i - \mathbf{x}^c)(\mathbf{x}_i - \mathbf{x}^c)^\top - \mathbf{P}_i \right) \mathbf{P}_i^{-1} \quad (5)$$

which should be negative definite (having all eigenvalues negative) at the mode location. If this condition is satisfied, all the sample points that converge to that location should be merged with a single Gaussian centered at the convergence location. Otherwise, they should be left unchanged since density approximation would create too high an error.

Figure 1 illustrates the convergence to the local maximum of each sample by the mode-finding algorithm. In each figure, the contour of a 2D density function constructed by kernel density estimation with 100 Gaussian kernels — all weights are equal in this example — is presented, and a white circle in Figure 1(a) indicates the initial location of each sample. As illustrated in the following figures, each sample converges to an associated mode and four modes are finally detected at the 14th time step. The mean-shift procedure is terminated when the size of the mean-shift vector is less than ϵ — a negligibly small number.

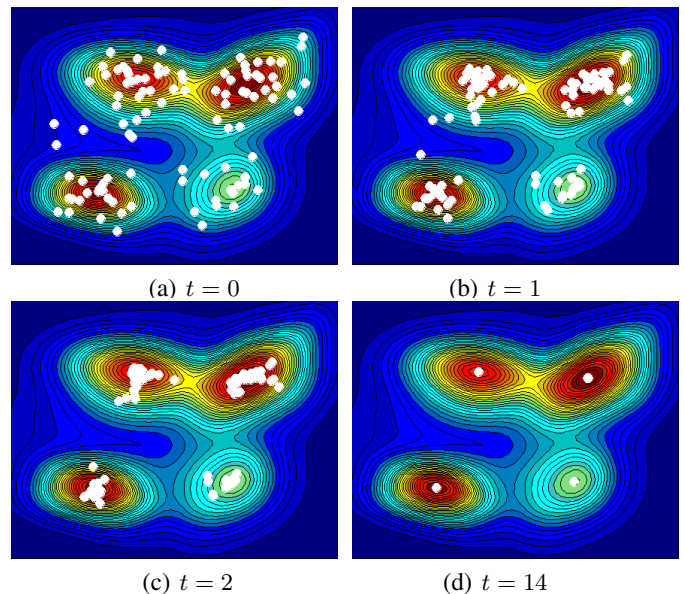


Fig. 1. Convergence by mode-finding algorithm

For density approximation, a Gaussian component is assigned to each detected mode, where the mean of the Gaussian is equal to the converged mode location and the weight of each Gaussian

is equal to the sum of the kernel weights of the data points that converge to the mode.

B. Covariance Estimation

The mean and weight of each Gaussian for the approximated density function are determined by mean-shift, and the covariance for each Gaussian need to be estimated. The covariance matrix associated with each mode $\tilde{\mathbf{P}}_j$ is computed by curvature fitting around the mode location using the Hessian matrix.

Suppose that the approximate density has m unique Gaussian component at $\tilde{\mathbf{x}}_j$ ($j = 1, \dots, m$) with associated weights $\tilde{\kappa}_j$ after the mode finding procedure. The Hessian matrix $\hat{\mathbf{H}}(\tilde{\mathbf{x}}_j)$ at a mode $\tilde{\mathbf{x}}_j$ of the original density function in equation (1) is given in equation (5). Also, the Hessian matrix at the mean of a Gaussian distribution centered at $\tilde{\mathbf{x}}_j$ with weight $\tilde{\kappa}_j$ and covariance $\tilde{\mathbf{P}}_j$ is given by

$$\mathbf{H}(\tilde{\mathbf{x}}_j) = -\frac{\tilde{\kappa}_j}{(2\pi)^{d/2} |\tilde{\mathbf{P}}_j|^{1/2}} \tilde{\mathbf{P}}_j^{-1}. \quad (6)$$

By equalizing these two expressions for the Hessian matrices at the mode in the original density and the single Gaussian distribution, we solve for the estimated covariance matrix $\tilde{\mathbf{P}}_j$. Specifically, suppose that $\tilde{\mathbf{P}}_j$ is decomposed by Singular Value Decomposition (SVD) as $\tilde{\mathbf{P}}_j = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$ and that the Hessian matrix at $\tilde{\mathbf{x}}_j$ in equation (5) is represented with a similar form, $\hat{\mathbf{H}}(\tilde{\mathbf{x}}_j) = \mathbf{V}\mathbf{\Gamma}\mathbf{V}^\top$. Then, the following equation is obtained by the equalization of two Hessian matrices.

$$\mathbf{V}\mathbf{\Gamma}\mathbf{V}^\top = -\frac{\tilde{\kappa}_j}{(2\pi)^{d/2} |\mathbf{\Lambda}|^{1/2}} \mathbf{U}^\top \mathbf{\Lambda}^{-1} \mathbf{U} \quad (7)$$

By assuming $\mathbf{U} = \mathbf{V}^\top$ and from equation (7), we can compute $|\mathbf{-\Gamma}|$ which is given by

$$|\mathbf{-\Gamma}| = -\frac{\tilde{\kappa}_j^d}{(2\pi)^{d^2/2}} |\mathbf{\Lambda}|^{-\frac{d+2}{2}}, \quad (8)$$

and $\mathbf{\Lambda}$ is derived from equation (7) and (8) as follows.

$$\mathbf{\Lambda} = -\frac{\tilde{\kappa}_j^{\frac{2}{d+2}}}{|2\pi(-\mathbf{\Gamma}^{-1})|^{\frac{1}{d+2}}} \mathbf{\Gamma}^{-1} \quad (9)$$

Therefore, the estimated covariance matrix is finally given by

$$\tilde{\mathbf{P}}_j = -\frac{\tilde{\kappa}_j^{\frac{2}{d+2}}}{|2\pi(-\hat{\mathbf{H}}_j^{-1}(\tilde{\mathbf{x}}_j))|^{\frac{1}{d+2}}} \hat{\mathbf{H}}_j^{-1}(\tilde{\mathbf{x}}_j), \quad (10)$$

and the approximated density is

$$\tilde{f}(\mathbf{x}) = \frac{1}{(2\pi)^{d/2}} \sum_{i=1}^m \frac{\tilde{\kappa}_i}{|\tilde{\mathbf{P}}_i|^{1/2}} \exp\left(-\frac{1}{2} D^2(\mathbf{x}, \tilde{\mathbf{x}}_i, \tilde{\mathbf{P}}_i)\right). \quad (11)$$

where $\tilde{\mathbf{P}}_i$ is given by equation (10) and $m(\ll n)$ is the number of detected modes.

C. Performance of Kernel Density Approximation

We next describe a set of simulations for testing the accuracy of kernel density approximation. We consider situations in which the underlying density is a mixture of Gaussian with an arbitrary number of components. The density function is reconstructed by kernel density estimation (KDE), EM algorithm and our kernel density approximation (KDA), and then the Mean Integrated

Squared Error (MISE) between the groundtruth and estimated densities are compared.

In our experiments, three different density functions are tested as specified in Table I and Figure 2; case 1 and 2 are relatively well-separated Gaussian mixtures, and the density function in case 3 involves non-symmetric modes and heavy-tailed regions. Note that a Gaussian distribution is denoted by $N(\cdot)$, which involves three parameters *weight*, *mean*, and *covariance*. The same bandwidth is used in kernel density estimation and kernel density approximation for each kernel, and the correct number of Gaussian components is given to the EM algorithm. The average MISE is computed based on 50 realizations for reliability.

As shown in Table I, kernel density approximation has comparable error with both kernel density estimation and EM. Since kernel density approximation assigns a Gaussian kernel to each detected mode and the covariance is estimated based on the curvature around each mode, the accuracy degrades in areas far from the mode locations. However, the density functions simulated by kernel density approximation are represented with only a small number of Gaussian components without any pre-defined parameters, and the error is still comparable to kernel density estimation and EM. On the other hand, the EM algorithm is powerful when the number of Gaussian components is known, but its performance is severely degraded by an incorrect setting of the number of components and/or inappropriate parameter initialization. Some examples of inaccurate density estimation by the EM algorithm are illustrated in Figure 3, where the reconstructed density functions incur high error mainly because of bad initialization of parameters. As seen in the figure, this problem is aggravated when the number of Gaussian components is wrong, as in Figure 3(a) and (c).

TABLE I
ERROR OF KDE AND KDA TO THE GROUNDTRUTH

MISE ($\times 10^{-5}$)	case 1	case 2	case 3
E_{kde}	2.2691	0.9387	0.9765
E_{kda}	2.6024	2.1003	3.7344
E_{em}	6.0664	2.0302	2.4773

- case 1: $N(0.15, 12, 5)$, $N(0.1, 50, 4)$, $N(0.35, 70, 8)$, $N(0.25, 90, 16)$, $N(0.15, 119, 32)$
- case 2: $N(0.15, 25, 10)$, $N(0.1, 37, 8)$, $N(0.15, 65, 16)$, $N(0.25, 77, 9)$, $N(0.15, 91, 30)$, $N(0.2, 154, 15)$
- case 3: $N(0.28, 100)$, $N(0.25, 30, 100)$, $N(0.15, 60, 64)$, $N(0.2, 100, 256)$, $N(0.2, 145, 576)$

A multi-dimensional example, which includes both well-separated Gaussian components as well as non-Gaussian areas, is shown in Figure 4; again, kernel density approximation results in reasonably accurate estimations with a small number of Gaussian components.

Kernel density approximation is a framework to estimate a density function with a mixture of Gaussians. Even though kernel density approximation provides accurate estimation with a small number of Gaussians and all relevant parameters are determined automatically, the approximation error is relatively high in areas where density modes are non-symmetric and/or there are heavy-tailed areas. This is because only a single Gaussian is assigned to each detected mode and the covariance for each Gaussian is estimated based only on the curvature around the mode. Kernel density approximation has one free parameter — kernel bandwidth, as does kernel density estimation. Fortunately, there

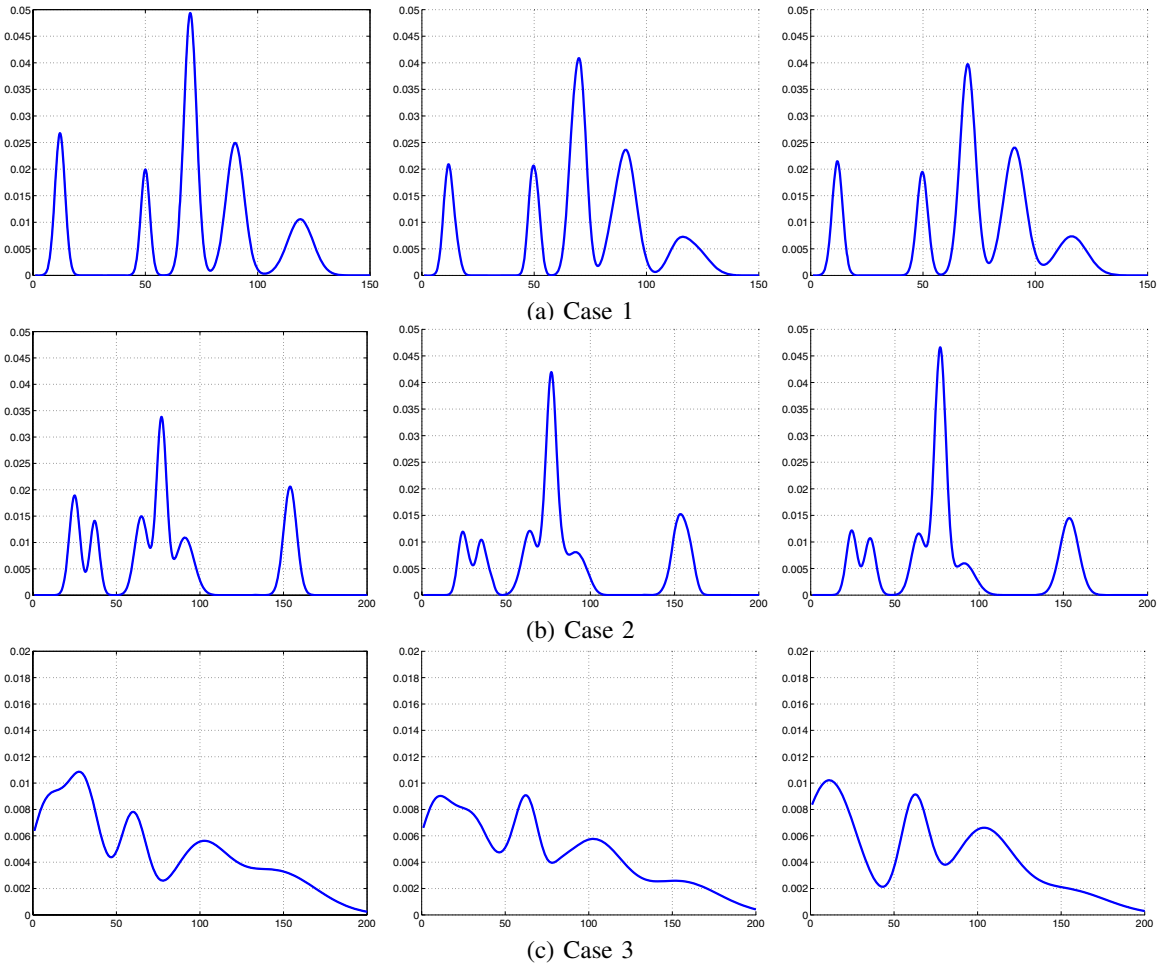


Fig. 2. Comparison between KDE and KDA in 1D. **(left)** Original density function **(middle)** KDE **(right)** KDA For the approximation, 200 samples are drawn from the original distribution

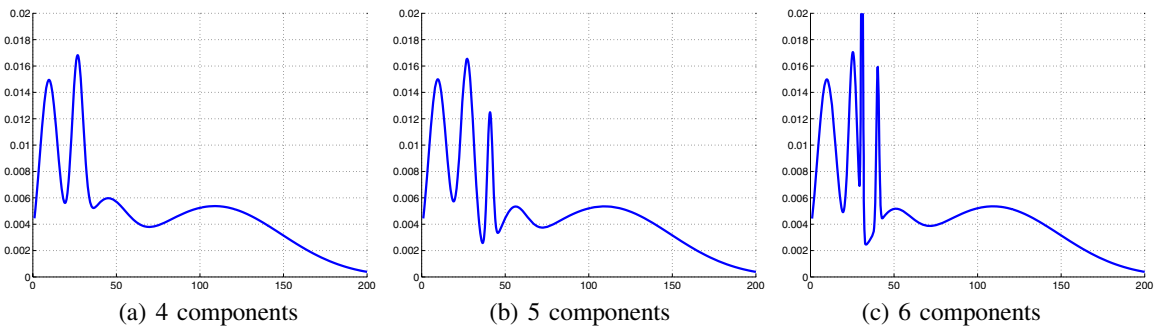


Fig. 3. Examples of bad approximation by EM algorithm (case 3 in Table I)

are several approaches to determine the kernel bandwidth as [1], [5]–[7] although no ideal solution for determining optimal bandwidth is known yet.

III. SEQUENTIAL KERNEL DENSITY APPROXIMATION

In many real-time computer vision applications, all the data is not initially available; instead, data is provided frame by frame. Therefore, on-line density estimation is needed for real-time processing. The procedure to update the density function is similar to the on-line EM algorithm [19], [23], but our method determines the number of Gaussian components in a more principled way at

each time step. In this section, we present a sequential version of kernel density approximation.

A. Naive Quadratic Time Algorithm

Assume that at time t the underlying density is a mixture of Gaussians having n_t modes and that for each mode we have allocated a Gaussian $N(\kappa_t^i, \mathbf{x}_t^i, \mathbf{P}_t^i)$, $i = 1, \dots, n_t$. For the moment, select a learning rate α and assume that all incoming data become part of the model. Let $N(\alpha, \mathbf{x}_{t+1}^{new}, \mathbf{P}_{t+1}^{new})$ be a new measurement. With the integration of the new measurement, the

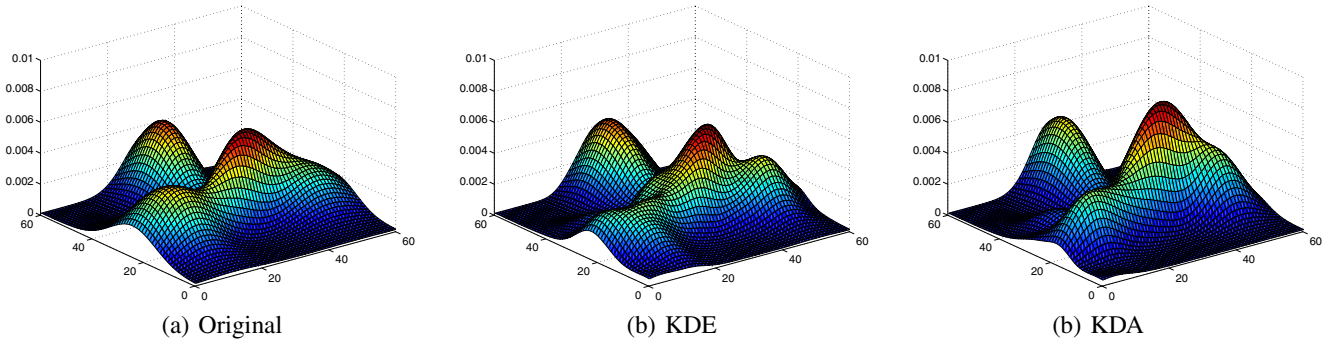


Fig. 4. Comparison between KDE and KDA with 200 samples in 2D. (a) KDE (MISE = 1.6684×10^{-6}) (b) KDA (MISE = 3.0781×10^{-6})

density at time $t + 1$ is initially written as

$$\hat{f}_{t+1}(\mathbf{x}) = \frac{(1-\alpha)}{(2\pi)^{d/2}} \sum_{i=1}^{n_t} \frac{\kappa_t^i}{|\mathbf{P}_t^i|^{1/2}} \exp\left(-\frac{1}{2}D^2(\mathbf{x}_t, \mathbf{x}_t^i, \mathbf{P}_t^i)\right) + \frac{\alpha}{(2\pi)^{d/2} |\mathbf{P}_{t+1}^{new}|^{1/2}} \exp\left(-\frac{1}{2}D^2(\mathbf{x}_t, \mathbf{x}_{t+1}^{new}, \mathbf{P}_{t+1}^{new})\right) \quad (12)$$

Starting now from locations \mathbf{x}_{t+1}^{new} and \mathbf{x}_t^i with $i = 1, \dots, n_t$, we perform mean shift iterations and let \mathbf{x}_{t+1}^{newc} and \mathbf{x}_t^{ic} , $i = 1, \dots, n_t$ be the convergence locations. We select first the convergence locations at which more than one \mathbf{x}_t^i or \mathbf{x}_{t+1}^{new} converged. Let \mathbf{y} be a point in this set and let \mathbf{x}^j , $j = 1, \dots, m$ be the starting locations for which the mean shift procedure converged to \mathbf{y} . If the Hessian $\hat{\mathbf{H}}(\mathbf{y}) = (\nabla \nabla^\top) \hat{f}_{t+1}(\mathbf{y})$ is negative definite, we associate with the mode \mathbf{y} a Gaussian component by $N(\kappa_y, \mathbf{y}, \mathbf{P}(\mathbf{y}))$ where κ_y is the sum of \mathbf{x}^j 's weights ($j = 1, \dots, m$) and the covariance matrix $\mathbf{P}(\mathbf{y})$ is determined by the method in section II-B. At time $t + 1$ the Gaussian components located at \mathbf{x}^j , $j = 1, \dots, m$ will be substituted for by a new Gaussian $N(\kappa_y, \mathbf{y}, \mathbf{P}(\mathbf{y}))$. In other words, those Gaussian components that converge to the same location after adding a new Gaussian are replaced by a single Gaussian component, so the number of components in the mixture does not increase indefinitely through the sequential update of the density function.

If the Hessian $\hat{\mathbf{H}}(\mathbf{y})$ is not negative definite (i.e., the location \mathbf{y} is either a saddle point or a local minimum), all the components associated with \mathbf{x}^j , $j = 1, \dots, m$ are left unchanged to avoid incurring too high an error.

For the convergence locations at which only one procedure converged (except the convergence location for \mathbf{x}_{t+1}^{new}), the weight, mean and covariance of the associated Gaussian component are also left unchanged.

B. Linear Time Algorithm for Sequential Approximation

The sequential kernel density approximation technique described in the previous section takes $O(n_t^2)$ time in each step, where n_t is the number of modes at time step t . Now, we relax the constraint that the number of Gaussian components is equal to the number of modes in the density function, and improve the time complexity to linear time. As a result, the number of Gaussian components may be slightly more than the compact representation introduced in section III-A, but the sequential kernel density approximation is performed much faster asymptotically.

Recall equation (12). The previous algorithm runs the mean-shift procedure for all of $n_t + 1$ components, and finds convergence

points for all of them. Then, it finds the mode associated with the new data, and updates that mode.¹ So, if we could efficiently identify those modes that would merge with the new data, then the execution time can be dramatically reduced. This technique is explained next.

1) *Algorithm Description*: Given the $n_t + 1$ modes in the $t + 1$ st step, we first search for the convergence point \mathbf{c}^{new} of \mathbf{x}_{t+1}^{new} in the density $\hat{f}_{t+1}(\mathbf{x})$ of equation (12). Now, we have to determine which other modes converge to \mathbf{c}^{new} and should be merged with \mathbf{x}_{t+1}^{new} . The candidates that converge to \mathbf{c}^{new} are determined by mean-shift, and this procedure is repeated until no additional candidate converges to \mathbf{c}^{new} . The first candidate mode is the convergence point \mathbf{x}_t^i ($i = 1, \dots, n_t$) of \mathbf{x}_{t+1}^{new} in the density function $\hat{f}_{t+1}(\mathbf{x}) = \hat{f}_{t+1}(\mathbf{x}) - N(\alpha, \mathbf{x}_{t+1}^{new}, \mathbf{P}_{t+1}^{new})$. Note that all the candidates are one of the components in the previous density function $\hat{f}_t(\mathbf{x})$. The mean-shift procedure is performed for \mathbf{x}_t^i in $\hat{f}_{t+1}(\mathbf{x})$, and we check if the convergence point of \mathbf{x}_t^i is equal to \mathbf{c}^{new} . If they are not equal, we conclude that there are no further merges with \mathbf{x}_{t+1}^{new} and create a Gaussian for the merged mode; otherwise, we check the next candidate which is determined by finding the next convergence point of \mathbf{x}_{t+1}^{new} in the density function $\hat{f}_{t+1}(\mathbf{x}) = \hat{f}_{t+1}(\mathbf{x}) - N(\kappa_t^i, \mathbf{x}_t^i, \mathbf{P}_t^i)$.

The covariance matrix and the weight of the merged mode should be also updated as described in section III-A. The formal description of this algorithm is given in Algorithm 1. In Algorithm 1, *MeanShiftModeFinding* is the function to detect the convergence location by the mean-shift algorithm from a point (the second argument) in the density (the first argument).

2) *Algorithm Analysis*: The time complexity of this algorithm is $O(n_{max})$ on average by amortized analysis, where n_{max} is the maximum number of modes in all time steps; a sketch of the proof is as follows.

Suppose that each new data element has $5n_{max} + 1$ credits, which is defined as the reserved number of operations for the Gaussian component corresponding to the new data. For the search for the convergence point (line 3 in Algorithm 1), at most $n_{max} + 1$ operations are performed and the new component consumes $n_{max} + 1$ credits since the function *MeanShiftModeFinding* takes linear time. $2n_{max}$ credits are required for two mean-shift iterations when the new component fails to merge (the last iteration of while loop). Also, we need $2n_{max}$ operations

¹Rarely, some merges which do not include the new data may happen. It hardly affects the accuracy of the density functions, but leads to a difference in the number of components between the quadratic and the linear time algorithm.

Algorithm 1 Linear-time sequential kernel density approximation

```

1:  $S = \{\mathbf{x}_{t+1}^{new}\}, \kappa = \alpha$ 
2:  $\hat{f}_{t+1}'(\mathbf{x}) = \hat{f}_{t+1}(\mathbf{x})$ 
3:  $\mathbf{c}^{new} = MeanShiftModeFinding(\hat{f}_{t+1}'(\mathbf{x}), \mathbf{x}_{t+1}^{new})$ 
4:  $\hat{f}_{t+1}'(\mathbf{x}) = \hat{f}_{t+1}'(\mathbf{x}) - N(\alpha, \mathbf{x}_{t+1}, \mathbf{P}_{t+1}^{new})$ 
5: while 1 do
6:    $\mathbf{x}_t^i = MeanShiftModeFinding(\hat{f}_{t+1}'(\mathbf{x}), \mathbf{x}_{t+1}^{new})$ 
7:    $\mathbf{c} = MeanShiftModeFinding(\hat{f}_{t+1}'(\mathbf{x}), \mathbf{x}_t^i)$ 
8:   if  $\mathbf{c}^{new} \neq \mathbf{c}$  then
9:     break
10:  end if
11:   $S = S \cup \{\mathbf{x}_t^i\}, \kappa = \kappa + \kappa_t^i$ 
12:   $\hat{f}_{t+1}'(\mathbf{x}) = \hat{f}_{t+1}'(\mathbf{x}) - N(\kappa_t^i, \mathbf{x}_t^i, \mathbf{P}_t^i)$ 
13: end while
14: merge all the modes in the set  $S$  and create  $N(\kappa, \mathbf{c}, \mathbf{P}_{\mathbf{c}})$  where
     $\mathbf{P}_{\mathbf{c}}$  is derived by the same method in equation (10)

```

(line 6 and 7) whenever the new component is merged with the currently existing mode, but the existing mode is responsible for this cost. So, the remaining $2n_{max}$ credits are supposed to be used when another mode is merged with it later. After losing all credits, the mode finally disappears.

For K time steps, K new Gaussian components are entered and sequential kernel density approximation is performed. Therefore, the number of operations for all K time steps is at most $O(Kn_{max})$, and the average time complexity in each time step is $O(n_{max})$. The derived complexity is bounded by $O(n_{max})$, but practically it is faster than this since the number of modes in each time step is less than n_{max} in most cases. The number of modes is slightly more than the previous quadratic time algorithm, but the improvement of time complexity is the dominating factor for the overall speed of algorithm.

C. Performance of Sequential Algorithm

The linear time sequential kernel density approximation algorithm, which we will refer to as the *fast approximation algorithm*, is a variant of the quadratic time algorithm. The performance of the fast approximation algorithm was tested through simulation, and compared with the quadratic time algorithm as well as a sequential version of kernel density estimation.

Starting from the initial density function, a new data element is incorporated at each step, and Mean Integrated Squared Error (MISE) with sequential kernel density estimation is employed as a basis of comparison. The weighted Gaussian mixture — $N(0.15, 80, 15^2)$, $N(0.4, 122, 10^2)$, and $N(0.45, 122, 100^2)$ — is used as the initial density function, and the new data is sampled from another Gaussian mixture — $N(0.2, 52, 10^2)$, $N(0.5, 100, 10^2)$, and $N(0.15, 175, 10^2)$ — plus a uniform distribution in $[0, 255]$ with weight 0.15. In this experiment, we expect the initial density function to morph to the new density function from which data samples are drawn.

As seen in Figure 5, the fast approximation algorithm accurately simulates the sequential kernel density estimation; the final density function has three major modes which closely correspond to the Gaussian centers in the sampling function. The simulated density function using the quadratic time algorithm is very similar to that of the fast approximation algorithm in most steps, so it is not presented separately in this figure. Note that the sequential

kernel density estimation has more than 300 Gaussian components at the 300th time step, but the fast approximation algorithm has only a small number of modes.

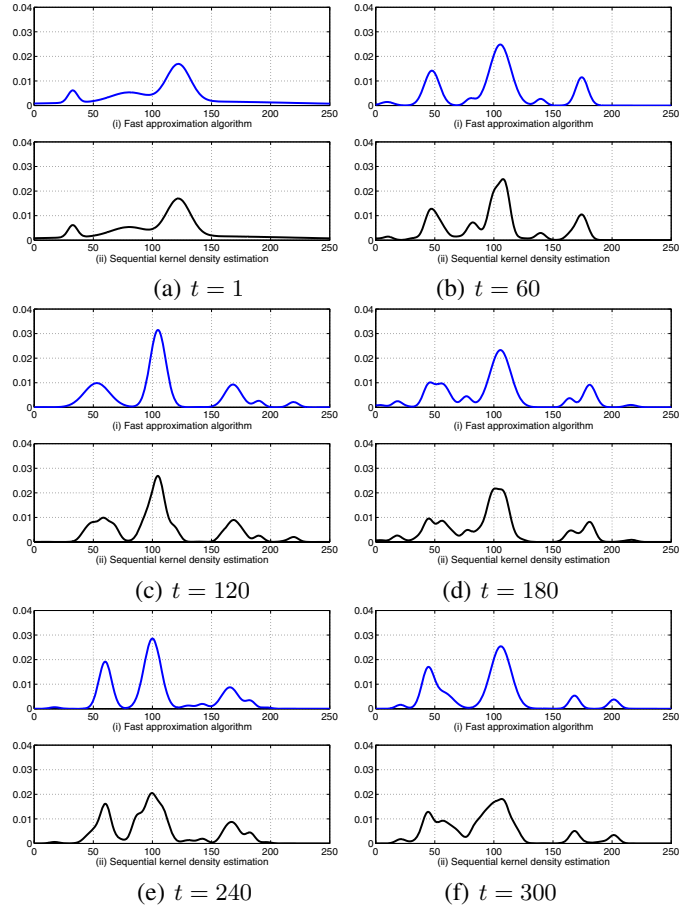


Fig. 5. Simulation of fast approximation algorithm and comparison with sequential kernel density estimation. (a)-(f) Fast sequential kernel density approximation (top) vs. sequential kernel density estimation (bottom).

Figure 6 shows that the MISE of the fast approximation algorithm is comparable to the quadratic time algorithm (and in repeated experiments, the new algorithm is often better) while the increase of the number of modes using the fast algorithm is moderate.

We also compared the density propagation by on-line EM algorithm [19], [23]. The initial density function is exactly the same, and a Gaussian mixture density function with three components is updated by the on-line EM algorithm at every time step. Note that the density function from which samples are drawn also has three components except uniform distribution component. As demonstrated in Figure 7, the Gaussian with the largest variance is not adapted well by new data, and significant error is observed around major modes. In this example, the on-line version of EM algorithm is not able to remove a component for the high variance distribution, nor to create a new component that is represented in the data.

The standard on-line EM algorithm does not update the number of components in the mixture, so, according to our experiment, the on-line EM algorithm is typically faster by two or three times than sequential kernel density approximation depending on the number of components in the mixture and the characteristics of

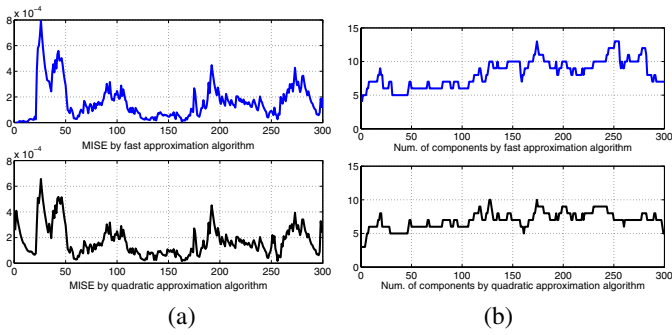


Fig. 6. Comparison of MISE and number of components in each step of 1D simulation between fast approximation algorithm (top) and quadratic time algorithm (bottom). (a) MISE (b) Number of modes

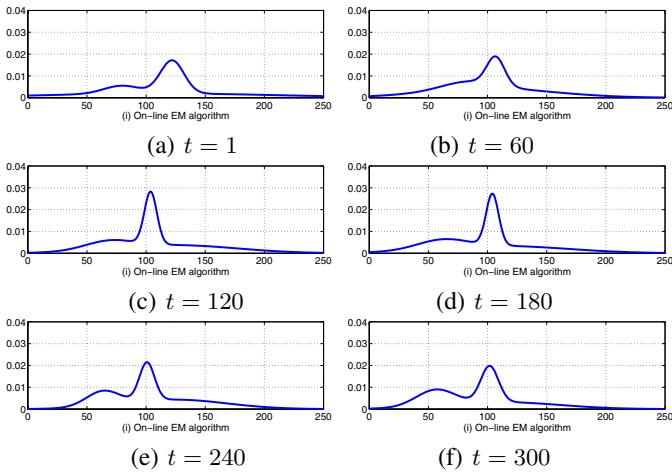


Fig. 7. Simulation of on-line EM algorithm.

the underlying density function.²

Multi-dimensional simulations were also performed, and it is observed that the major modes of density functions are preserved well although there are relatively high errors in areas far from these modes. Figures 8 and 9 present the simulation results, showing the comparisons of MISE and number of modes, respectively.

To compare the speed of the linear and quadratic time algorithms, the CPU time for the one-step sequential density approximation procedure was measured for density functions with different numbers of components. Since sequential density estimation contains matrix operations, it is also worthwhile to check the performance with respect to dimensionality. So, our experiments are performed by varying the number of modes and the dimensionality. We performed comparisons for two different dimensionalities (2D and 6D), and the results are presented in Figure 10. We observe that the running time of the fast approximation algorithm is significantly less than the quadratic time algorithm.

Finally, we performed 100-step sequential density estimations in various dimensions, and computed the average CPU time and MISE³. Figure 11 illustrates that the fast approximation algorithm is faster and comparable in accuracy.

In the next section, we describe how to apply the sequential

²Unfortunately, one-to-one speed comparison is not straightforward due to different update strategies for the number of components.

³The MISE is computed only at sample locations in this case to handle high dimensional examples.

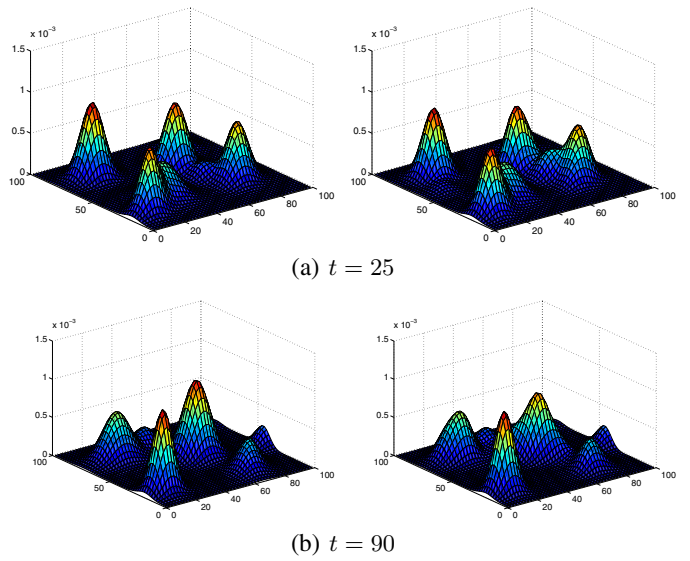


Fig. 8. Simulation of fast sequential kernel density approximation (left) and comparison with sequential kernel density estimation (right)

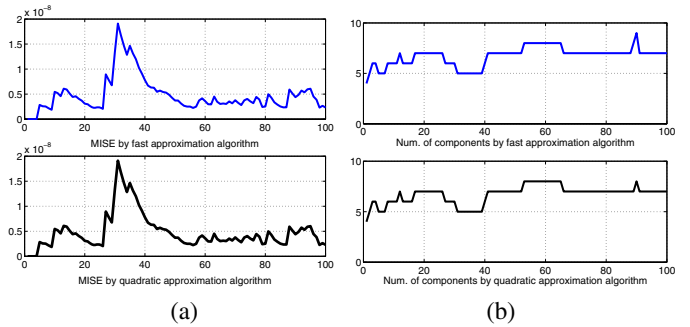


Fig. 9. Comparison of MISE and number of modes in each step of 2D simulation between fast approximation algorithm (top) and quadratic time algorithm (bottom). (a) MISE (b) Number of modes

kernel density approximation technique to on-line target appearance modeling for visual tracking.

IV. ON-LINE TARGET APPEARANCE MODELING FOR OBJECT TRACKING

Real-time object tracking is a challenging task. One of the greatest challenges to creating robust trackers is the construction of adaptive appearance models which can accommodate unstable lighting condition, pose variations, scale changes, view-point changes, and camera noise. Many tracking algorithms [2]–[4], [8], [26] are based on a fixed target model, which makes it difficult to apply them over long time intervals because of target appearance changes.

Some efforts have been made to overcome these problems. In [21], heuristics regarding the replacement of the target template are suggested; Nummiaro et al. [25] update the model by taking the weighted average of the current and new histograms. Recently, Ross et al. [28] propose an adaptive tracking algorithm that updates the models using an incremental update of eigenbasis. Instead of using a template or a histogram for target modeling, parametric density representations have been used in many tracking algorithms. A Gaussian distribution and its update by a Kalman filter for target template is proposed in [24]; a Gaussian

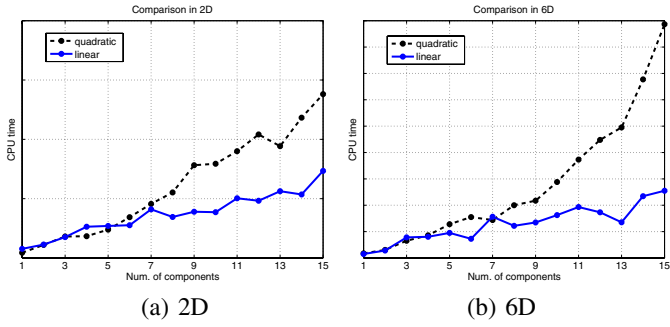


Fig. 10. CPU time of fast approximation algorithm and quadratic time algorithm.

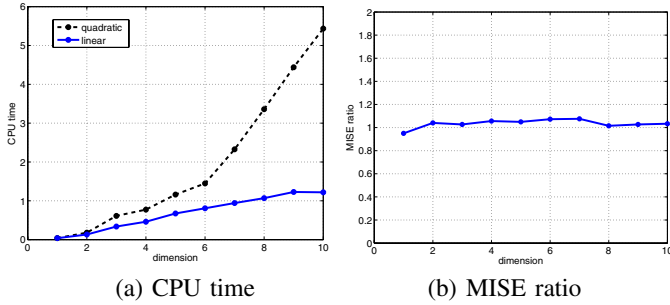


Fig. 11. CPU time and MISE with respect to dimensionality in sequential kernel density approximation. (MISE ratio is error ratio of fast approximation algorithm to quadratic time algorithm.)

distribution is utilized to model the target template, and it is updated by a Kalman filter. McKenna et al. [22] suggest using a Gaussian mixture model computed by an EM algorithm, but their method requires knowledge of the number of components, which may be difficult to know in advance. number of modes changes frequently. A more elaborate target model is described in [19], where a 3-component mixture for the stable process, the outlier data and the wandering term is designed to capture rapid temporal variations in the model. However, the implementation of that method also requires a fixed number of components. Additionally, it cannot accommodate multiple stable components.

The most important issue in target model update is the balance between adaptiveness to new observations and resistance to noise. Since the target model can drift away by undesirable updates, only target pixel observations should be integrated into the model. From this point of view, a probability density function of visual features is a good solution for target modeling, because frequently observed data contribute the most significant part while outliers can have limited effects on the integrity of the model.

In this section, we present a density based target modeling and on-line model update algorithm to deal with changes in target appearance, and compare its performance with other methods.

A. Target Modeling

We construct a model for each pixel in the target object with a mixture of Gaussians, so the target model is represented as a set of density functions. Our representation can include an arbitrary number of Gaussian components in each density function, and describes the underlying density accurately.

Using pixel-wise color density modeling has the advantage of describing the details of a target region, but does not capture any

spatial aspects of an object’s appearance. So, we also incorporate rectangular features into our target model. These features are obtained by averaging the intensities of neighbors (e.g., 3×3 or 5×5) in each color channel (r, g, b) and are computed efficiently with *integral images* [31]. Since rectangular features encode the spatial information around a pixel, they ameliorate some problems caused by non-rigid motions of objects and pixel misregistrations. The performance of such features have previously been investigated in object tracking [13] and detection [31].

Initially, at time 0, the density function for each pixel (i, j) within a selected target region has a single Gaussian component $N(1, \mathbf{x}_0(i, j), \mathbf{P}_0(i, j))$ whose mean $\mathbf{x}_0(i, j)$ is the combination of color at (i, j) and average color of its neighborhood. In each time step t , the new data $\mathbf{x}_t(i, j)$ at the pixel location (i, j) is denoted as

$$\mathbf{x}_t(i, j) = (r, g, b, \bar{r}, \bar{g}, \bar{b}) \quad (13)$$

where $(\bar{r}, \bar{g}, \bar{b})$ denotes the average intensity of the neighborhood centered at (i, j) for each color channel. Note that $\mathbf{x}_t(i, j)$ would be a 2D vector composed of intensity and average intensity for gray scale images.

Whenever a new observation is integrated into the current density, the density function is updated as explained in Section III-B. As time progresses, highly weighted Gaussian components are constructed around frequently observed data, and several minor modes may also develop. Using exponential updating, old information is removed gradually if no further observations are nearby. So, the representation maintains a history of feature observations for any given pixel.

B. Update in Scale

The target model so constructed is robust to changes of feature values, but is not intended to handle scale change. Tracking may fail in sequences containing large scale changes of target objects, since the observations are severely affected by drastic up- or down-sampling.

The scale in each frame is determined in tracking algorithm, and we update the size of the target model at every $\beta\%$ scale change as follows: the pixel location in the new target window is projected into the old target window, and the density function is computed by a weighted sum of neighborhood density functions as

$$\hat{f}_{\mathbf{x}}^*(i, j) = \sum_{(u, v) \in A(i, j)} w(u, v) \hat{f}_{\mathbf{x}}(u, v) \quad (14)$$

where $A(i, j)$ is a set of pixels adjacent to (i, j) ’s projection onto the old target window, $\hat{f}_{\mathbf{x}}(u, v)$ is an estimated density function for feature vector \mathbf{x} at (u, v) , and $w(u, v)$ is the normalized weight associated with each density function $\hat{f}_{\mathbf{x}}(u, v)$. The new density function $\hat{f}_{\mathbf{x}}^*(i, j)$ is also a mixture of Gaussians, and the kernel density approximation technique presented in section II is applied to reduce the number of components for a compact representation.

The modeling error in scale change is fairly small because of the spatial coherence of the target. Also, since the rectangular features for adjacent pixels are based on highly overlapping areas, they are robust to updates in scale. This appearance model update in scale is simple, but performs well in experiments; the strategy plays an important role in the examples displaying significant scale change.

C. Maximum Likelihood Parameter Optimization

Any general tracking algorithm can be utilized with our target appearance modeling method; we adopted a simple gradient-based method in our experiments.

Let $\mathbf{M}(\mathbf{x}; \mathbf{p})$ denote the parameterized motion of location \mathbf{x} , where $\mathbf{p} = (v_x, v_y, s)$ is a vector for translation and scale. Denote by $d(\mathbf{M}(\mathbf{x}; \mathbf{p}))$ the observation at the image of \mathbf{x} under motion \mathbf{p} . Now, the tracking problem involves finding the optimal parameter using the maximum likelihood method.

$$\mathbf{p}_t = \arg \max_{\mathbf{p}} \sum_{(i,j) \in R} p(d(\mathbf{M}(\mathbf{x}^{(i,j)}; \mathbf{p})) | \mathbf{p}_{t-1}, A_t^{(i,j)}) \quad (15)$$

where (i, j) is the relative location in the target region R , and $A_t^{(i,j)}$ the current appearance model at (i, j) . The local gradient of likelihood in the parameter space is computed, and the optimal parameter is obtained by iteratively moving in the gradient-ascent direction until convergence. Once the optimal location and scale of the target is found, the target model is updated by the method described in Section IV-A if the current target and model template size changes by less than $\beta\%$; otherwise the update in scale presented in Section IV-B should be performed, additionally.

D. Experiments

Various sequences are tested to illustrate our on-line appearance modeling technique. New data is integrated with the weight of $\alpha = 0.05$. Also, the target model is updated in scale at every $\beta = 10\%$ change of size, and a slightly higher learning rate $\alpha = 0.1$ is used at this time. For the rectangular features, 5×5 neighborhood pixels are used.

The learning rate is critical to overall performance. It would be ideal to adjust the learning rate based on the confidence of observations and the prediction of target appearance changes. However, this is a very difficult problem by itself, so we use fixed values, which are determined empirically. The tracking results presented are unaffected by minor changes to the learning rate — $3 \sim 5\%$ (without scale update) and $5 \sim 10\%$ (with scale update). This is because our sequential kernel density approximation algorithm is effectively adapts to the consistent data and is resistant to noise. Also, the kernel bandwidth for new data is fixed for the entire sequence in the following examples, since the overall tracking performance is almost the same for a large range of kernel bandwidth according to both [11] and our own experiments.

Figure 12 shows the results on a *tank* sequence, in which the target has low contrast and changes its orientation during tracking. Our tracking algorithm succeeds in tracking for the entire 940 frames even with transient severe noise levels due to dust (e.g., Figure 12 (b) and (d)).

The results on a *person* sequence are presented next. In this sequence, a human face is tracked with a large change of face orientation and lighting conditions; the results are shown in Figure 13. Figure 13 (e) illustrates how the appearance of the face changes over time, where the color of each pixel is determined by weighted mean of a Gaussian mixture associated with the pixel.⁴ Figure 13 (f) shows the average intensity of the target region over time, and it suggests that brightness changes significantly during tracking and it is difficult to track accurately without adaptive appearance modeling.

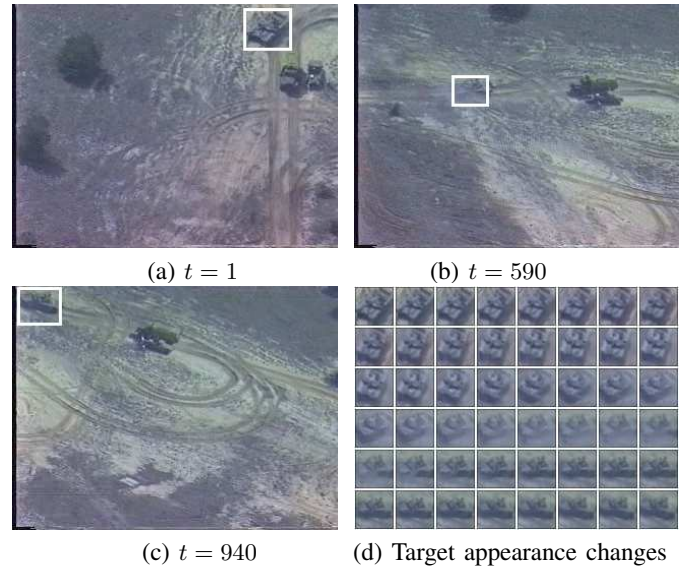


Fig. 12. Tracking result of *tank* sequence

In Figure 14, the tracking results for the head of a football player are presented. In this sequence, the target object moves fast in high clutter and is blurred frequently. Also, the changes in orientation and texture of the head make tracking difficult, so the density function for the target model must be able to accommodate those variations for accurate tracking. As shown in Figure 14 (f), the average number of components per pixel (solid line) is moderate but the variation (dotted line) is high, which suggests that a density function with a fixed number of components may produce a high tracking error since some pixels require a large number of components for accurate modeling.

In Figure 15, a gray scale image sequence involving a large scale change is presented, and our appearance model update strategy adapts to this well.

In the last sequence, the appearance of the car changes frequently because of the shadow and its red lights. Also, a person passes in front of the car and the image is severely shaken several times by camera movement. Figure 16 shows that our on-line density-based modeling is successful in this sequence in spite of occlusion and appearance change. Figure 17 illustrates internal information of tracker for the frame at $t = 18$. Figure 17(a) presents the local similarity map centered at the target location at the previous time step (white triangle), where the highest likelihood location (black triangle) means the new target location.⁵ On the other hand, target window, target appearance model and pixel likelihood map are depicted in Figure 17(b). In the pixel likelihood map, the bright pixels indicate high likelihood. As shown in the figure, the occluded area has very low similarity with the target appearance model and low likelihood, so has very limited effect in finding the optimal location.

Two different appearance modeling methods are implemented for comparison; one is fixed modeling with a Gaussian distribution and the other is modeling by a Gaussian mixture with three components which is update by an on-line EM algorithm. For each algorithm, two different feature sets — color only and color with rectangular features for each pixel — are employed. Using

⁴The rectangular features are not used in the visualization.

⁵The optimal target size does not change in this frame, so the scale for current target size is selected for visualization.

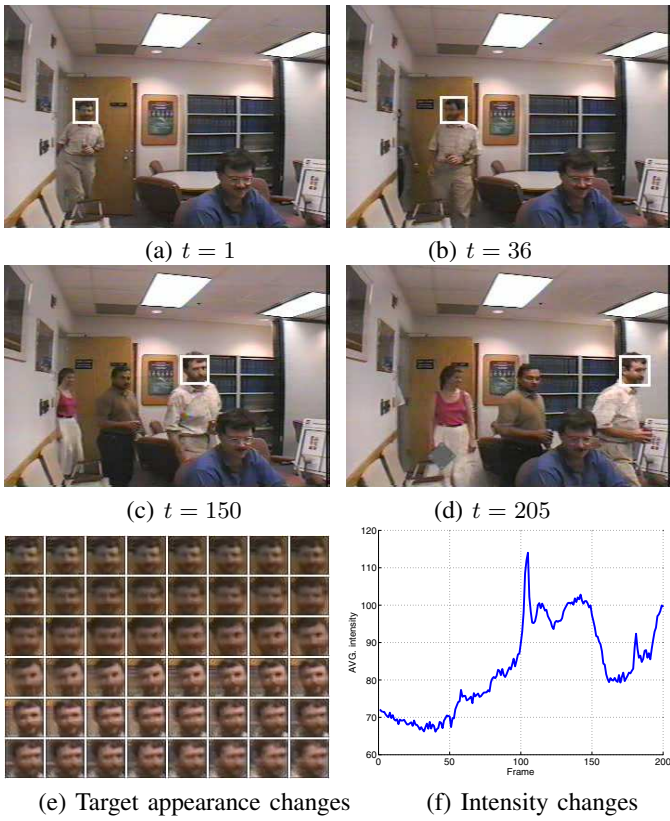


Fig. 13. Tracking results of *person* sequence

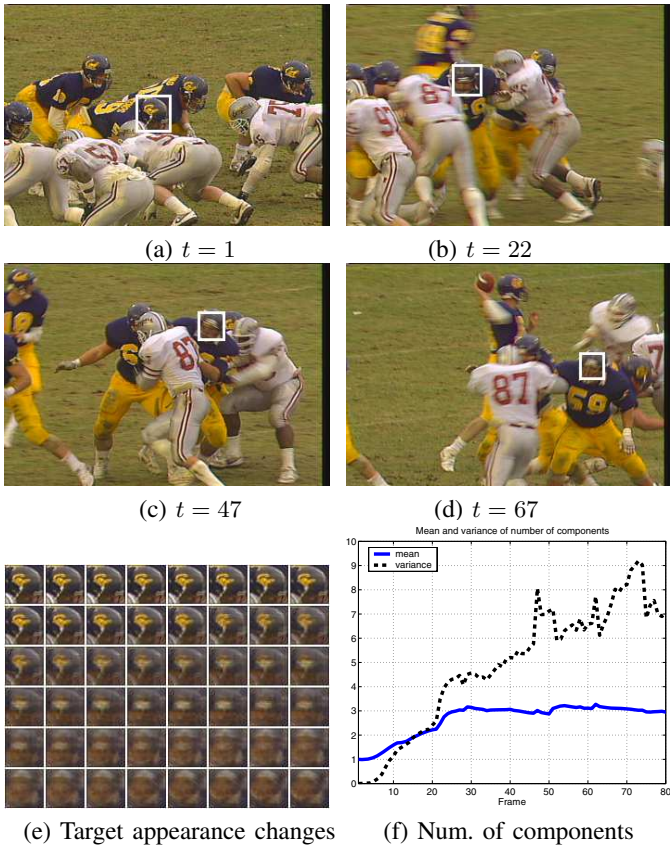


Fig. 14. Tracking result of *football* sequence

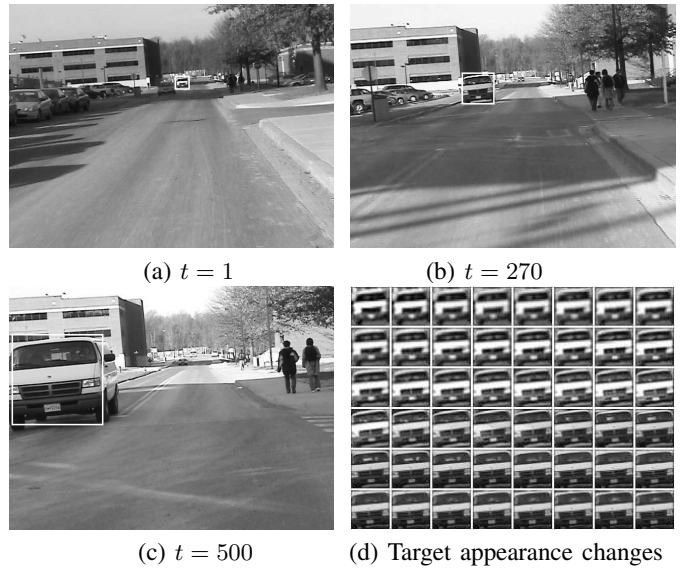


Fig. 15. Tracking result of *car1* sequence

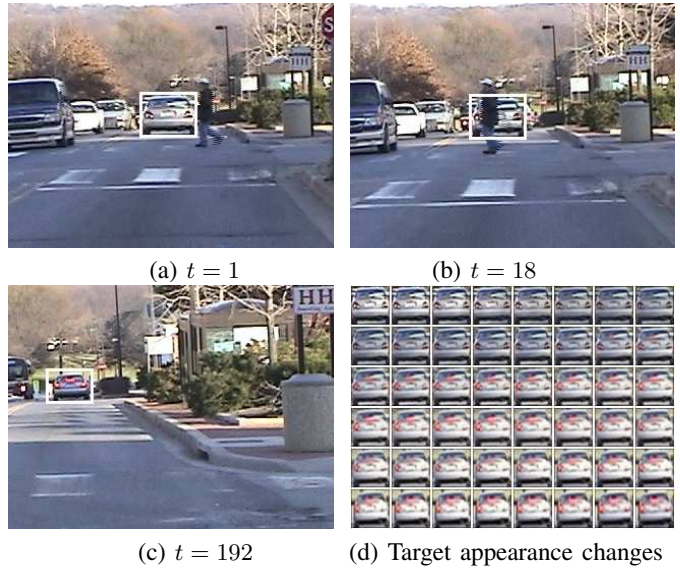


Fig. 16. Tracking result of *car2* sequence

the same gradient-based tracking, six different cases including two for our algorithm are tested, and the results are summarized in table II. The numbers in the table indicate the rough frame numbers where tracking fails. Also, some examples of failures are illustrated in Figure 18. As table II illustrates, our on-line density approximation shows good performance compared with other parametric techniques. The only algorithm able to successfully track through both sequences was our method, using both pixel-wise and rectangular features.

V. CONCLUSION

We described kernel density approximation and its sequential update strategy in which the density function is represented with a mixture of Gaussians. This representation is advantageous over conventional parametric or non-parametric techniques since every parameter for a Gaussian mixture — number, means, covariances and weights for each Gaussian component — is automatically

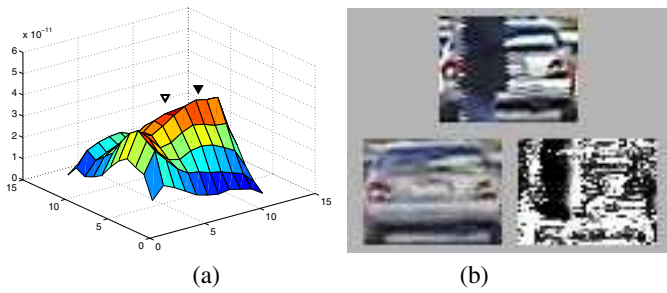


Fig. 17. Supplementary data for the frame at $t = 18$ of *car2* sequence. (a) local similarity map (b) target window (top), appearance model (lower left) and pixel similarity (lower right)

TABLE II
 COMPARISON OF TRACKING RESULTS

Modeling method		<i>person</i> (205 frames)	<i>football</i> (80 frames)
Fixed	C ^a	fail (110)	fail (20)
Gaussian	C+R ^b	fail (110)	fail (20)
Gaussian mixture	C	succeed	fail (55)
	C+R	succeed	fail (25)
Our method	C	succeed	fail (70)
	C+R	succeed	succeed

^acolor feature

^bcolor rectangular feature

determined. The sequential update procedure has linear time complexity, and the sequential kernel density approximation technique can be incorporated in many real-time applications with low computational cost. The use of the new framework was illustrated in on-line target appearance modeling for object tracking. The performance of kernel density approximation was illustrated by various simulations and experiments with synthetic examples and natural videos.

ACKNOWLEDGEMENT

This research was funded in part by the U.S. Government's VACE program. The first author thanks Yoo-Ah Kim in the University of Connecticut for the valuable comments.

REFERENCES

[1] I. Abramson, "On bandwidth variation in kernel estimates - a square root law," *The Annals of Statistics*, vol. 10, no. 4, pp. 1217-1223, 1982.
 [2] M. J. Black and A. Jepson, "Eigentracking: Robust matching and tracking of articulated objects using a view-based representation," in *Proc. European Conf. on Computer Vision*, Cambridge, UK, 1996, pp. 610-619.
 [3] T. Cham and J. Rehg, "A multiple hypothesis approach to figure tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Fort Collins, CO, volume II, 1999, pp. 239-249.
 [4] H. Chen and T. Liu, "Trust-region methods for real-time tracking," in *Proc. 8th Intl. Conf. on Computer Vision*, Vancouver, Canada, volume II, 2001, pp. 717-722.
 [5] W. Cleveland, "Robust locally weighted regression and smoothing scatterplots," *J. Amer. Statist. Assn.*, vol. 74, pp. 829-836, 1979.
 [6] W. Cleveland and C. Loader, "Smoothing by local regression: Principles and methods," *Statistical Theory and Computational Aspects of Smoothing*, pp. 10-49, 1996.
 [7] D. Comaniciu, "An algorithm for data-driven bandwidth selection," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 25, no. 2, pp. 281-288, 2003.
 [8] D. Comaniciu, V. Ramesh, and P. Meer, "Real-time tracking of non-rigid objects using mean shift," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Hilton Head, SC, volume II, June 2000, pp. 142-149.



Fig. 18. Examples of tracking failure. Tracking for *person* sequence with the fixed Gaussian modeling (C+R, $t = 120$) (left) and for *football* sequence with the 3-component mixture modeling (C, $t = 59$) (right)

[9] D. Comaniciu, V. Ramesh, and P. Meer, "The variable bandwidth mean shift and data-driven scale selection," in *Proc. 8th Intl. Conf. on Computer Vision*, Vancouver, Canada, volume I, July 2001, pp. 438-445.
 [10] T. Cormen, C. Leiserson, and R. Rivest, *Introduction to algorithms*. MIT Press and McGraw-Hill, 1990.
 [11] A. Elgammal, R. Duraiswami, and L. Davis, "Probability tracking in joint feature-spatial spaces," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Madison, Wisconsin, June 2003.
 [12] A. Elgammal, R. Duraiswami, D. Harwood, and L. Davis, "Background and foreground modeling using nonparametric kernel density estimation for visual surveillance," *Proceedings of IEEE*, vol. 90, pp. 1151-1163, 2002.
 [13] P. Fieguth and D. Teropoulos, "Color-based tracking of heads and other mobile objects at video frame rates," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, San Juan, Puerto Rico, 1997, pp. 21-27.
 [14] B. Frey, "Filling in scenes by propagating probabilities through layers into appearance models," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Hilton Head, SC, volume I, 2000, pp. 185-192.
 [15] B. Han, D. Comaniciu, and L. Davis, "Sequential kernel density approximation through mode propagation: Applications to background modeling," in *Asian Conference on Computer Vision* Jeju Island, Korea, 2004.
 [16] B. Han and L. Davis, "On-line density-based appearance modeling for object tracking," in *Proc. 10th Intl. Conf. on Computer Vision*, Beijing, China, 2005.
 [17] I. Haritaoglu, D. Harwood, and L. Davis, "W4: Real-time surveillance of people and their activities," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 809-830, 2000.
 [18] A. Jepson and M. Black, "Mixture models for optical flow computation," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, New York, 1993, pp. 760-761.
 [19] A. Jepson, D. Fleet, and T. El-Maraghi, "Robust online appearance models for visual tracking," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Hawaii, volume I, 2001, pp. 415-422.
 [20] D. Lee, "Effective gaussian mixture learning for video background subtraction," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 27, no. 5, pp. 827-832, 2005.
 [21] I. Matthews, T. Ishikawa, and S. Baker, "The template update problem," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 26, no. 6, pp. 810-815, 2004.
 [22] S. McKenna, Y. Raja, and S. Gong, "Tracking colour objects using adaptive mixture models," *Image and Vision Computing Journal*, vol. 17, pp. 223-229, 1999.
 [23] R. Neal and G. Hinton, "A view of the EM algorithm that justifies incremental, sparse, and other variants," in *Learning in Graphical Models*, M.I. Jordan, ed., 1998, pp. 355-368.
 [24] H. T. Nguyen, M. Worring, and R. van den Boomgaard, "Occlusion robust adaptive template tracking," in *Proc. 8th Intl. Conf. on Computer Vision*, Vancouver, Canada, October 2001.
 [25] K. Nummiaro, E. Koller-Meier, and L. V. Gool, "An adaptive color-based particle filter," *Image and Vision Computing*, vol. 21, no. 1, pp. 99-110, 2003.
 [26] P. Perez, C. Hue, J. Vermaak, and M. Gangnet, "Color-based probabilistic tracking," in *Proc. European Conf. on Computer Vision*, Copenhagen, Denmark, volume I, 2002, pp. 661-675.
 [27] C. Priebe and D. Marchette, "Adaptive mixture density estimation," *Pattern Recog.*, vol. 26, no. 5, pp. 771-785, 1993.
 [28] D. Ross, J. Lim, and M. Yang, "Adaptive probabilistic visual tracking with incremental subspace update," in *Proc. European Conf. on Computer Vision*, Prague, Czech, volume II, May 2004, pp. 470-482.

- [29] C. Stauffer and W. Grimson, "Learning patterns of activity using real-time tracking," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 22, no. 8, pp. 747–757, 2000.
- [30] O. Tuzel, F. Porikli, and P. Meer, "A bayesian approach to background modeling," in *Proc. Intl. Workshop on Machine Vision for Intelligent Vehicles in conjunction with CVPR*, San Diego, CA, 2005.
- [31] P. Viola and M. Jones, "Rapid object detection using a boosted cascade of simple features," in *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, Kauai, Hawaii, 2001, pp. 511–518.
- [32] C. Wren, A. Azarbayejani, T. Darrell, and A. Pentland, "Pfinder: Real-time tracking of the human body," *IEEE Trans. Pattern Anal. Machine Intell.*, vol. 19, pp. 780–785, 1997.
- [33] C. Yang, R. Duraiswami, N. Gumerov, and L. Davis, "Improved fast gauss transform and efficient kernel density estimation," in *Proc. 9th Intl. Conf. on Computer Vision*, Nice, France, volume I, 2003, pp. 464–471.



Ying Zhu received the Ph.D. degree from Princeton University, Princeton, NJ, in 2003.

Since 2003, she has been with the Real-Time Vision and Modeling Department of Siemens Corporate Research, Inc., Princeton. Her research interests include statistical methods for computer vision, real-time object detection, tracking and recognition, statistical learning, sensor fusion, and image processing. With Siemens Corporate Research, she has developed vision-based object detection and tracking techniques for driver assistance and video surveil-

lance applications.

Dr. Zhu was the Co-organizer of the first International Workshop on Machine Vision for Intelligent Vehicles (MVIV'05) in conjunction with the IEEE International Conference on Computer Vision and Pattern Recognition in 2005. She is a member of the IEEE.



Bohyung Han received B.S. and M.S. degree from Department of Computer Engineering in Seoul National University, Seoul, Korea, in 1997 and 2000, respectively. Also he received the Ph.D. degree in Department of Computer Science from the University of Maryland – College Park, MD, in 2005.

He was a senior research engineer in Samsung Electronics R&D Center in Irvine, CA from 2006 to 2007, and now a research scientist in the University of California at Irvine.

His research interests include statistical analysis in computer vision, object detection and tracking, sensor fusion, machine learning, image processing, multimedia and bioinformatics. He is a member of the IEEE.



Dorin Comaniciu Dorin Comaniciu received the Ph.D. degrees in electrical engineering from the Polytechnic University of Bucharest in 1995 and from Rutgers University in 1999.

Since 1999, he has been with Siemens Corporate Research in Princeton, first as a member of technical staff, then as a senior member of technical staff and manager of the Statistical Methods for Vision Systems Program. He is currently the head of the Integrated Data Systems Department. His research interests include robust methods for computer vision,

motion estimation, nonparametric analysis, robust information fusion, medical imaging, biomedical informatics, content-based access to visual data, and integrated information modeling. He holds 18 US patents and has co-authored more than 120 papers, conference papers, and book chapters in the area of visual information processing. For his work in object tracking, he has received the Best Paper Award at the 2000 IEEE Conference on Computer Vision and Pattern Recognition. For his innovations in the areas of medical imaging and intelligent vehicles, he has received the 2004 Siemens Inventor of the Year Award, the highest technical recognition of Siemens AG worldwide.

He serves as an Associate Editor of IEEE Transactions on Pattern Analysis and Machine Intelligence and was an Associate Editor of Pattern Analysis and Applications between 2002 and 2004. He leads the scientific direction of one of the largest European projects in biomedical informatics, Health-e-Child. He is a senior member of the IEEE.



Larry S. Davis received his B.A. from Colgate University in 1970 and his M. S. and Ph. D. in Computer Science from the University of Maryland in 1974 and 1976 respectively. From 1977-1981 he was an Assistant Professor in the Department of Computer Science at the University of Texas, Austin. He returned to the University of Maryland as an Associate Professor in 1981. From 1985-1994 he was the Director of the University of Maryland Institute for Advanced Computer Studies. He is currently a Professor in the Institute and the Computer Science

Department, as well as Chair of the Computer Science Department. He was named a Fellow of the IEEE in 1997.

Prof. Davis is known for his research in computer vision and high performance computing. He has published over 100 papers in journals and has supervised over 20 Ph.D. students. He is an Associate Editor of the International Journal of Computer Vision and an area editor for Computer Models for Image Processing: Image Understanding. He has served as program or general chair for most of the field's major conferences and workshops, including the 5th International Conference on Computer Vision, the 2004 Computer Vision and Pattern Recognition Conference, and the 11th International Conference on Computer Vision.